Reducing nonlinear dynamical systems via model reduction and machine learning

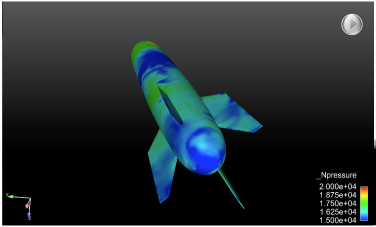
Kevin Carlberg

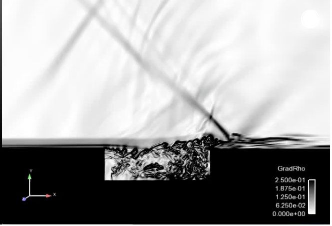
Sandia National Laboratories
Livermore, California

Uncertainty Quantification and Data-Driven Modeling
Austin, Texas
March 24, 2017

Goal: break computational barrier

High-fidelity computational models





+ Validated RANS/LES model: matches experiment to within 5%
- *Large scale*: 86 million cells; 200,000 time steps
- *High simulation costs*: 6 weeks; 5000 cores

Barrier

Many query applications

- Uncertainty quantification
- Design optimization

Nonlinear dynamical systems and many-query problems

Full-order model (FOM)

Full-order model
ODE

↓

time discretization

↓

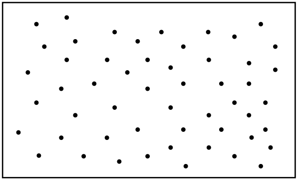
Full-order model
OΔE

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}; t, \boldsymbol{\mu});\ \boldsymbol{x}(0, \boldsymbol{\mu}) = \boldsymbol{x}^0(\boldsymbol{\mu}),\ t \in [0, T],\ \boldsymbol{\mu} \in \mathcal{D}$$

$$\boldsymbol{r}^n(\boldsymbol{x}^n; \boldsymbol{\mu}) = 0, \quad n = 1, \ldots, N, \quad \boldsymbol{\mu} \in \mathcal{D}$$

Many-query problems

Goal: compute QoI $q(\boldsymbol{x}^n; \mu)$, $n = 1, \ldots, N$ for $\mu \in \mathcal{D}_{\text{eval}} \subset \mathcal{D}$



$\mathcal{D}_{\mathrm{eval}}$

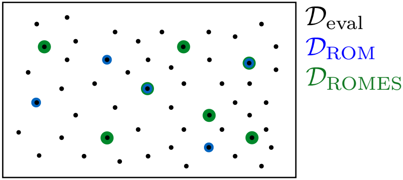
This is intractable with a large-scale FOM

Approach: ROM and ROMES

Reduce the FOM dimensionality and
quantify the introduced uncertainty

1. Reduced-order model (ROM)
   - **Goal**: low-dim dynamical system that accurately represents FOM
   - **Approach**: unsupervised machine learning and projection
   - \+ physics-based approximation
   - \+ can preserve special problem structure
   - \+ high speedups possible
2. Reduced-order model error surrogate (ROMES)
   - **Goal**: unbiased, low-variance statistical model of the ROM error
   - **Approach**: supervised machine learning (regression)
   - \+ more useful than error bounds (overpredict)
   - \+ quantifies ROM-induced epistemic uncertainty
   - \+ enables rigorous integration with UQ

Approach: leverage simulation data

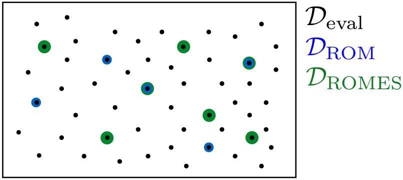


Offline:

1. **ROM training**: solve FOM for $\mu \in \mathcal{D}_{\text{ROM}} \subset \mathcal{D}_{\text{eval}}$
   - State and residual snapshots
2. **ROM construction**
   - *Unsupervised ML*: discover structure in ROM training data
   - *Projection*: reduce FOM dimensionality
3. **ROMES training**: solve ROM and FOM for $\mu \in \mathcal{D}_{\text{ROMES}} \subseteq \mathcal{D}_{\text{eval}}$
   - ROM error indicators
   - ROM QoI error
4. **ROMES construction**
   - *Supervised ML*: map ROM error indicators to ROM QoI error

Online: solve ROM + ROMES for remaining points in $\mathcal{D}_{\text{eval}}$

Approach: leverage simulation data



Offline:

1. **ROM training**: solve FOM for $\mu \in \mathcal{D}_{\text{ROM}} \subset \mathcal{D}_{\text{eval}}$
   - State and residual snapshots
2. **ROM construction**
   - *Unsupervised ML*: discover structure in ROM training data
   - *Projection*: reduce FOM dimension
3. **ROMES training**
4. **ROMES construction**

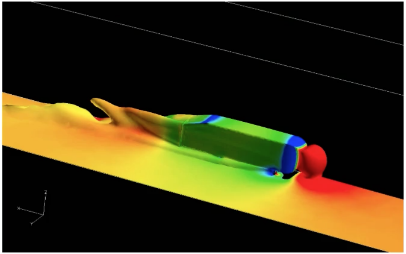
Online: solve ROM + ROMES for remaining points in $\mathcal{D}_{\text{eval}}$

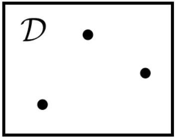
Collaborators: M. Barone (Sandia), H. Antil (GMU)

ROM training

$$\boldsymbol{r}^n(\boldsymbol{x}^n;\boldsymbol{\mu}) = 0, \quad n = 1, \ldots, N, \quad \boldsymbol{\mu} \in \mathcal{D}_{\mathrm{ROM}}$$

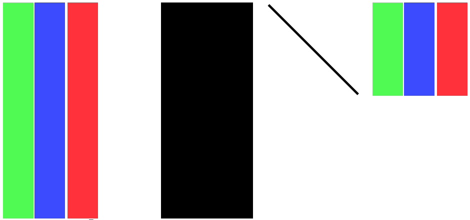
1 Collect 'snapshots' of the state (and residual)





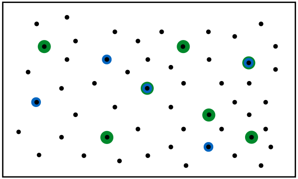
ROM construction: unsupervised machine learning

- Principal component analysis (i.e., POD)
  - Compute SVD: $[\,\mathbf{X}_1\ \mathbf{X}_2\ \mathbf{X}_3\,] = \mathbf{U}\ \boldsymbol{\Sigma}\ \mathbf{V}^T$



  - Truncate: $\boldsymbol{\Phi} = [\boldsymbol{u}_1\ \cdots\ \boldsymbol{u}_p]$
  - Repeat for residual to construct $\boldsymbol{\Phi}_R$
- Clustering
  - Construct sampling matrix $\boldsymbol{P}$ from residual data [C. et al., 2013]

Approach: leverage simulation data



Offline:

1. **ROM training**: solve FOM for $\mu \in \mathcal{D}_{\mathsf{ROM}} \subset \mathcal{D}_{\mathsf{eval}}$
   - State and residual snapshots
2. **ROM construction**
   - *Unsupervised ML*: discover structure in ROM training data
   - *Projection*: reduce FOM dimension
3. **ROMES training**
4. **ROMES construction**

Online: solve ROM + ROMES for remaining points in $\mathcal{D}_{\mathsf{eval}}$

How to perform projection with state basis $\mathbf{\Phi}$?

Full-order model
ODE

↓

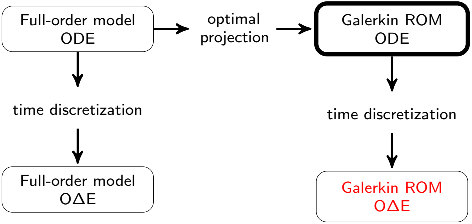
time discretization

↓

Full-order model
OΔE

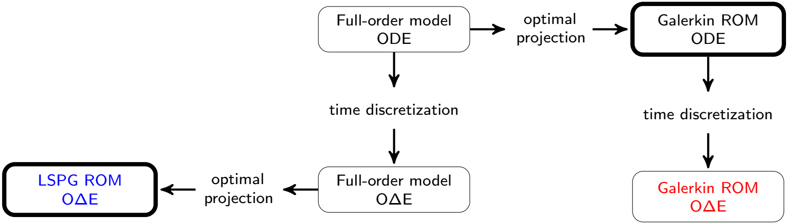
How to perform projection with state basis $\mathbf{\Phi}$?

- Optimize then discretize? (common)



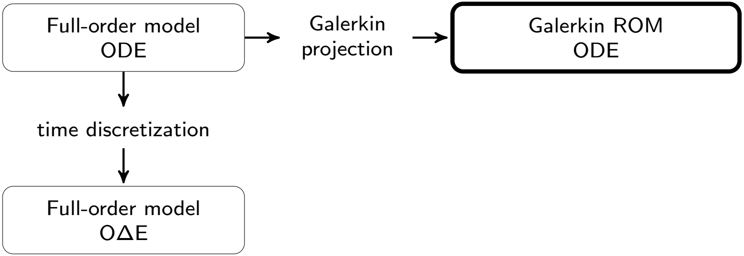
How to perform projection with state basis $\mathbf{\Phi}$?

- Optimize then discretize? (common)
- Discretize then optimize? (uncommon)



Comparative analysis: C, Barone, Antil, "Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction," Journal of Computational Physics, 330:693–734, 2017.

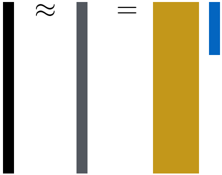
Galerkin ROM: first optimize



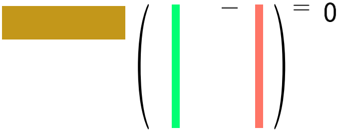
Galerkin ROM

- ODE: Galerkin projection on FOM ODE

1 $\boldsymbol{x}(t;\boldsymbol{\mu}) \approx \tilde{\boldsymbol{x}}(t;\boldsymbol{\mu}) = \boldsymbol{\Phi}\hat{\boldsymbol{x}}(t;\boldsymbol{\mu})$



2 $\boldsymbol{\Phi}^T\left(\boldsymbol{f}(\tilde{\boldsymbol{x}}, t;\boldsymbol{\mu}) - \frac{d\tilde{\boldsymbol{x}}}{dt}\right) = 0$



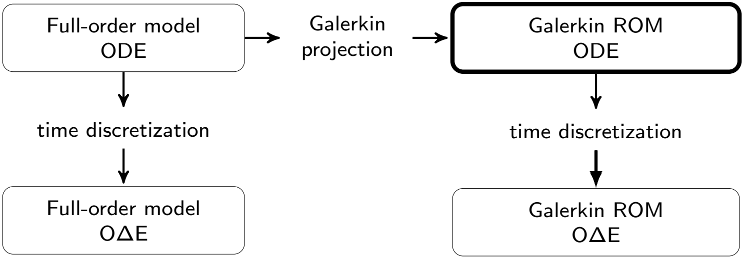
$$\frac{d\hat{\boldsymbol{x}}}{dt} = \boldsymbol{\Phi}^T\boldsymbol{f}(\boldsymbol{\Phi}\hat{\boldsymbol{x}}, t;\boldsymbol{\mu}), \quad \hat{\boldsymbol{x}}(0;\boldsymbol{\mu}) = \boldsymbol{\Phi}^T\boldsymbol{x}^0(\boldsymbol{\mu}), \quad t \in [0, T], \quad \boldsymbol{\mu} \in \mathcal{D}$$

Theorem (Galerkin ROM: time-continuous optimality)

The Galerkin ROM velocity minimizes the time-continuous FOM residual:

$$\frac{d\tilde{\boldsymbol{x}}}{dt}(\boldsymbol{\Phi}\hat{\boldsymbol{x}}, t;\boldsymbol{\mu}) = \arg\min_{\boldsymbol{v}\in range(\boldsymbol{\Phi})} \|\boldsymbol{v} - \boldsymbol{f}(\boldsymbol{\Phi}\hat{\boldsymbol{x}}, t;\boldsymbol{\mu})\|_2^2.$$

Galerkin: first optimize, then discretize



Galerkin ROM

- ODE

$$\frac{d\hat{\boldsymbol{x}}}{dt} = \boldsymbol{\Phi}^T \boldsymbol{f}(\boldsymbol{\Phi}\hat{\boldsymbol{x}}, t; \boldsymbol{\mu}), \quad \hat{\boldsymbol{x}}(0) = \boldsymbol{\Phi}^T \boldsymbol{x}^0(\boldsymbol{\mu}), \quad t \in [0, T], \quad \boldsymbol{\mu} \in \mathcal{D}$$

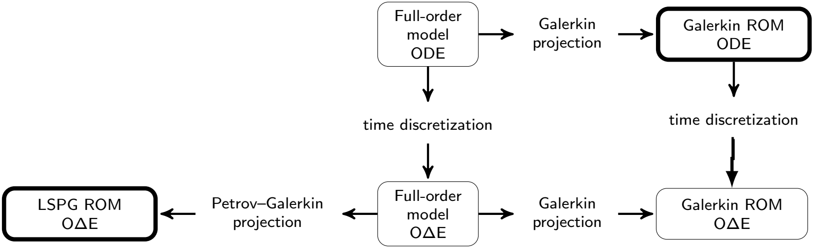
  + Continuous velocity $\frac{d\hat{\boldsymbol{x}}}{dt}$ is optimal

- OΔE

$$\boldsymbol{\Phi}^T \boldsymbol{r}^n (\boldsymbol{\Phi}\hat{\boldsymbol{x}}^n; \boldsymbol{\mu}) = 0, \quad n = 1, \ldots, N, \quad \boldsymbol{\mu} \in \mathcal{D}$$

  - Discrete state $\hat{\boldsymbol{x}}^n$ is not generally optimal

LSPG ROM: first discretize, then optimize



LSPG ROM

- FOM OΔE

$$\boldsymbol{r}^n(\boldsymbol{x}^n;\boldsymbol{\mu}) = 0, \quad n = 1, \ldots, N, \quad \boldsymbol{\mu} \in \mathcal{D}$$

- LSPG ROM OΔE:

$$\hat{\boldsymbol{x}}^n = \arg\min_{\hat{\boldsymbol{z}} \in \mathbb{R}^p} \|\boldsymbol{A}\boldsymbol{r}^n(\boldsymbol{\Phi}\hat{\boldsymbol{z}};\boldsymbol{\mu})\|_2^2, \quad n = 1, \ldots, N, \quad \boldsymbol{\mu} \in \mathcal{D}$$

$$\Updownarrow$$

$$\boldsymbol{\Psi}^n(\hat{\boldsymbol{x}}^n;\boldsymbol{\mu})^T\boldsymbol{r}^n(\boldsymbol{\Phi}\hat{\boldsymbol{x}}^n;\boldsymbol{\mu}) = 0, \quad n = 1, \ldots, N, \quad \boldsymbol{\mu} \in \mathcal{D}$$

  - $\boldsymbol{\Psi}^n(\hat{\boldsymbol{x}};\boldsymbol{\mu}) := \boldsymbol{A}^T\boldsymbol{A}\frac{\partial \boldsymbol{r}^n}{\partial \boldsymbol{x}}(\boldsymbol{\Phi}\hat{\boldsymbol{x}};\boldsymbol{\mu})$

  \+ Discrete solution is optimal

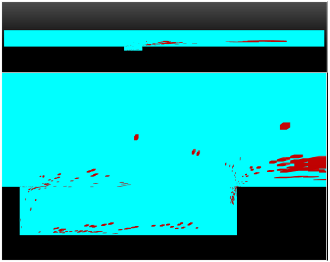
How to select weighting matrix $\boldsymbol{A}$? [C. et al., 2013]

$$\hat{\boldsymbol{x}}^n = \arg\min_{\hat{\boldsymbol{z}}\in\mathbb{R}^p} \|\boldsymbol{A}\boldsymbol{r}^n(\boldsymbol{\Phi}\hat{\boldsymbol{z}};\mu)\|_2^2$$

- Gappy POD approx of residual $\boldsymbol{r}^n \approx \tilde{\boldsymbol{r}}^n = \boldsymbol{\Phi}_R(\boldsymbol{P}\boldsymbol{\Phi}_R)^+\boldsymbol{P}\boldsymbol{r}^n$

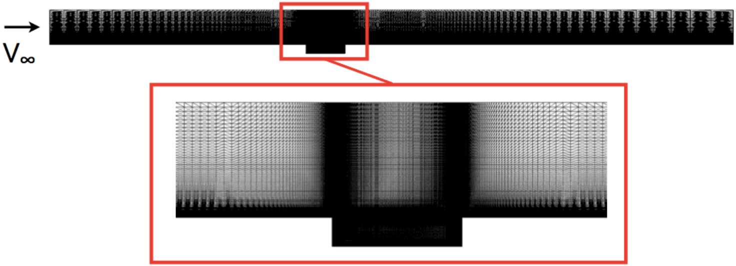
$$\hat{\boldsymbol{x}}^n = \arg\min_{\hat{\boldsymbol{z}}\in\mathbb{R}^p} \|\tilde{\boldsymbol{r}}^n(\boldsymbol{\Phi}\hat{\boldsymbol{z}})\|_2^2 \Leftrightarrow \hat{\boldsymbol{x}}^n = \arg\min_{\hat{\boldsymbol{z}}\in\mathbb{R}^p} \|\underbrace{(\boldsymbol{P}\boldsymbol{\Phi}_R)^+\boldsymbol{P}}_{\boldsymbol{A}_{\mathrm{GNAT}}}\boldsymbol{r}^n(\boldsymbol{\Phi}\hat{\boldsymbol{z}})\|_2^2$$

- *Sample mesh*: Extract mesh subset needed to compute $\boldsymbol{P}\boldsymbol{r}^n$



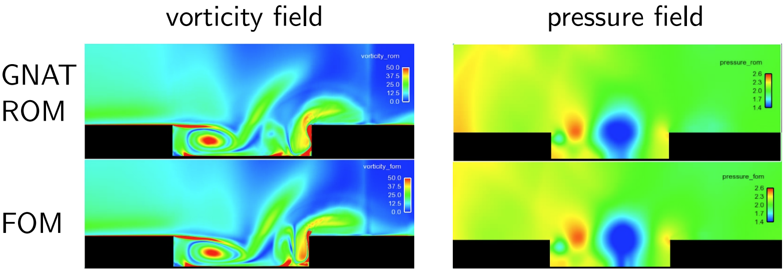
\+ Small problem size: can run on many fewer cores

Cavity-flow problem Collaborator: M. Barone (SNL)



- Unsteady, compressible Navier–Stokes
- DES turbulence model
- $M_\infty = 0.6$
- $\text{Re} = 6.3 \times 10^6$
- 1.2×10^6 degrees of freedom

GNAT performance ($t \leq 12.5$ sec)



+ $< 1\%$ error in time-averaged drag
+ Sample mesh: 4.1% nodes, 3.0% cells
+ 229x CPU-hour savings
  - FOM: 5 hour x 48 CPU
  - GNAT ROM: 32 min x 2 CPU
- Galerkin unstable

Why is LSPG more accurate than Galerkin? [C. et al., 2017]

Theorem (Local *a posteriori* bounds: BDF schemes)

If the following conditions hold:

1. $\exists \kappa > 0$ *such that* $\|\boldsymbol{f}(\boldsymbol{x}, \cdot; \cdot) - \boldsymbol{f}(\boldsymbol{y}, \cdot; \cdot)\|_2 \leq \kappa \|\boldsymbol{x} - \boldsymbol{y}\|_2$, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$
2. Δt *small enough such that* $0 < h := |\alpha_0| - |\beta_0| \kappa \Delta t$
3. *A BDF scheme is employed for time integration, then*

$$\|\delta \boldsymbol{x}_G^n\| \leq \frac{1}{h} \|\boldsymbol{r}_G^n(\boldsymbol{\Phi} \hat{\boldsymbol{x}}_G^n; \boldsymbol{\mu})\|_2 + \frac{1}{h} \sum_{\ell=1}^{k} |\alpha_\ell| \|\delta \boldsymbol{x}_G^{n-\ell}\|$$

$$\|\delta \boldsymbol{x}_L^n\| \leq \frac{1}{h} \min_{\boldsymbol{y} \in range(\boldsymbol{\Phi})} \|\boldsymbol{r}_P^n(\boldsymbol{y}; \boldsymbol{\mu})\|_2 + \frac{1}{h} \sum_{\ell=1}^{k} |\alpha_\ell| \|\delta \boldsymbol{x}_L^{n-\ell}\|$$

- $\delta \boldsymbol{x}_G^n := \boldsymbol{x}_\star^n - \boldsymbol{\Phi} \hat{\boldsymbol{x}}_G^n$.
- $\delta \boldsymbol{x}_L^n := \boldsymbol{x}_\star^n - \boldsymbol{\Phi} \hat{\boldsymbol{x}}_L^n$

LSPG sequentially minimizes the time-local error bound

Can we use this bound for error estimation?

Time-global error bound [C. et al., 2017]

Theorem (Global *a posteriori* bounds: BDF schemes)

If the following conditions hold:

1. $\exists \kappa > 0$ *such that* $\|\boldsymbol{f}(\boldsymbol{x},\cdot;\cdot) - \boldsymbol{f}(\boldsymbol{y},\cdot;\cdot)\|_2 \leq \kappa \|\boldsymbol{x} - \boldsymbol{y}\|_2$, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$
2. Δt *small enough such that* $0 < h := |\alpha_0| - |\beta_0|\kappa\Delta t$
3. *A BDF scheme is employed for time integration, then*

$$\|\delta \boldsymbol{x}_G^n\| \leq \frac{\gamma_1(\gamma_2)^n \exp(\gamma_3 t^n)}{\gamma_4 + \gamma_5 \Delta t} \max_{j\in\{1,\ldots,n\}} \|\boldsymbol{r}_G^j(\boldsymbol{\Phi}\hat{\boldsymbol{x}}_G^j;\boldsymbol{\mu})\|_2$$

$$\|\delta \boldsymbol{x}_L^n\| \leq \frac{\gamma_1(\gamma_2)^n \exp(\gamma_3 t^n)}{\gamma_4 + \gamma_5 \Delta t} \max_{j\in\{1,\ldots,n\}} \min_{\boldsymbol{y}\in range(\boldsymbol{\Phi})} \|\boldsymbol{r}_P^j(\boldsymbol{y};\boldsymbol{\mu})\|_2$$

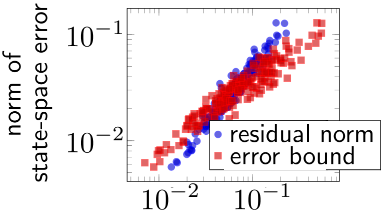
- $\delta \boldsymbol{x}_G^n := \boldsymbol{x}_\star^n - \boldsymbol{\Phi}\hat{\boldsymbol{x}}_G^n$.
- $\delta \boldsymbol{x}_L^n := \boldsymbol{x}_\star^n - \boldsymbol{\Phi}\hat{\boldsymbol{x}}_L^n$

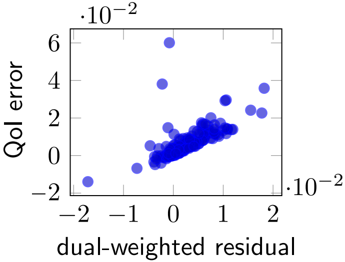
Global error bounds grow exponentially in time and overpredict the error

Deterministic: not amenable to integration with UQ

Idea: *construct accurate statistical error estimates from data*

Observation: ROMs generate error indicators that inform the error

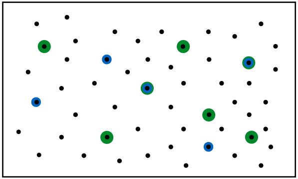




Goal: map error indicators (features) to the ROM error (response)

1. High-dimensional regression model (supervised ML)
   - maps error indicators to a prediction of the error
   - **methods**: random forest, support vector machine, k-NN
   - \+ enables many candidate error indicators to be considered
2. Gaussian-process model
   - maps regression-model output to a distribution over the error
   - \+ removes regression-model bias
   - \+ GP variance quantifies the ROM-induced epistemic uncertainty

Approach: leverage simulation data



Offline:

1. **ROM training**
2. **ROM construction**
3. **ROMES training**: solve ROM and FOM for $\mu \in \mathcal{D}_{\text{ROMES}} \subseteq \mathcal{D}_{\text{eval}}$
   - ROM error indicators
   - ROM QoI error
4. **ROMES construction**
   - *Supervised ML*: map ROM error indicators to ROM QoI error

Online: solve ROM + ROMES for remaining points in $\mathcal{D}_{\text{eval}}$

Collaborators: M. Drohmann, B. Freno (Sandia);
S. Trehan, L. Durlofsky (Stanford)

ROMES formulation [Drohmann and C., 2015, Trehan et al., 2017]

- FOM produces sequence of QoI values

$$\boldsymbol{\mu} \mapsto q_{\mathrm{FOM}}^n(\boldsymbol{\mu}) := q(\boldsymbol{x}^n(\boldsymbol{\mu}); \boldsymbol{\mu}), \quad n = 1, \dots, N$$

- ROM: produces sequence of QoI and error-indicator values

$$\boldsymbol{\mu} \mapsto q_{\mathrm{ROM}}^n := q(\boldsymbol{\Phi}\boldsymbol{x}^n(\boldsymbol{\mu}); \boldsymbol{\mu}), \quad n = 1, \dots, N$$
$$\boldsymbol{\mu} \mapsto \boldsymbol{\rho}^n(\boldsymbol{\mu}), \quad n = 1, \dots, N$$

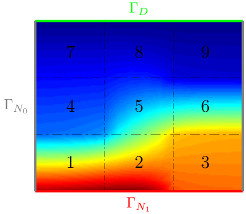
ROMES training:

1. Solve ROM and FOM for $\mu \in \mathcal{D}_{\mathrm{ROMES}}$
2. Training data: $\{(\boldsymbol{\rho}^n(\boldsymbol{\mu}), q_{\mathrm{FOM}}^n(\boldsymbol{\mu}) - q_{\mathrm{ROM}}^n(\boldsymbol{\mu})\}_{\boldsymbol{\mu}\in\mathcal{D}_{\mathrm{ROMES}}}$

ROMES construction:

1. Apply supervised ML to predict response from features
   - **Features**: error indicators $\boldsymbol{\rho}^n(\boldsymbol{\mu})$
   - **Response**: error $q_{\mathrm{FOM}}^n(\boldsymbol{\mu}) - q_{\mathrm{ROM}}^n(\boldsymbol{\mu})$
2. GP postprocessing to remove bias and quantify variance

Example 1: GP only, stationary problem [Drohmann and C., 2015]

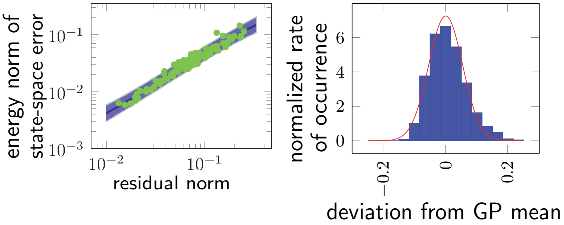


$$\triangle c(x;\boldsymbol{\mu})u(x;\boldsymbol{\mu}) = 0 \text{ in } \Omega \qquad \boldsymbol{x}(\boldsymbol{\mu}) = 0 \text{ on } \Gamma_D$$

$$\nabla c(\boldsymbol{\mu})\boldsymbol{x}(\boldsymbol{\mu}) \cdot n = 0 \text{ on } \Gamma_{N_0} \qquad \nabla c(\boldsymbol{\mu})\boldsymbol{x}(\boldsymbol{\mu}) \cdot n = 1 \text{ on } \Gamma_{N_1}$$

- **Inputs**: $\boldsymbol{\mu} \in [0.1, 10]^9$ define diffusivity c in subdomains
- **ROM**: RB–Greedy [Patera and Rozza, 2006]

Error: energy norm of state-space error
Error indicator: residual norm

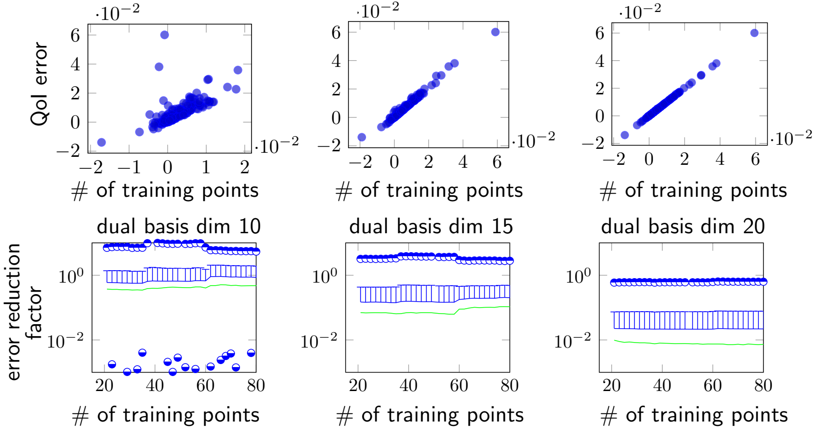


+ Unbiased, low-variance model of the error
+ Numerically validated
- Error bound overprediction as high as 8.0

Error: error in temperature at a point

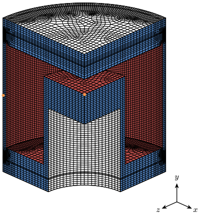
Error indicator: dual-weighted residual

$$\hat{\boldsymbol{y}}(\boldsymbol{\mu})^T \boldsymbol{Y}^T \boldsymbol{r}(\boldsymbol{\Phi}\hat{\boldsymbol{x}};\boldsymbol{\mu}) \text{ with } \boldsymbol{Y}^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}(\boldsymbol{\Phi}\hat{\boldsymbol{x}};\boldsymbol{\mu})^T \boldsymbol{Y}\hat{\boldsymbol{y}}(\boldsymbol{\mu}) = -\boldsymbol{Y}^T \frac{\partial q}{\partial \boldsymbol{x}}(\boldsymbol{\Phi}\hat{\boldsymbol{x}};\boldsymbol{\mu})$$



+ Uncertainty control: lower variance as columns added to $\boldsymbol{Y}$
+ Error can be reduced by up to two orders of magnitude

Example 2: ML and GP, stationary problem [Freno and C, 2017]

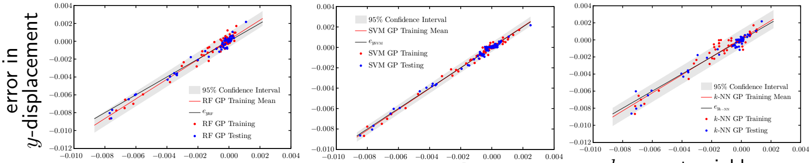


Predictive Capability Assessment Project (PCAP)

- Mechanical response
- 2.8×10^5 degrees of freedom
- **Inputs**: $\boldsymbol{\mu} \in [50\text{ GPa}, 100\text{ GPa}] \times [0.2,\ 0.35]$ define tube elastic modulus and Poisson ratio
- **QoI**: displacement of node of interest (orange)
- **ROM**: POD–Galerkin with $|\mathcal{D}_{\mathrm{ROM}}| = 8$
- **ROMES**: 150 data points ($|\mathcal{D}_{\mathrm{ROMES}}| = 30$ and five ROM basis dimensions)

Error: error in y-displacement at a point

Error indicators: 5000 elements of residual, input parameters

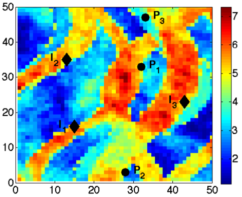


random forest error prediction | support vector machine error prediction | k-nearest neighbors error prediction

+ ML methods yield low-variance error predictions
+ ML methods amenable to large number of error indicators
+ Gaussian process removes regression-model bias

Example #3: ML and GP, nonlinear dynamical system

[Trehan et al., 2017]

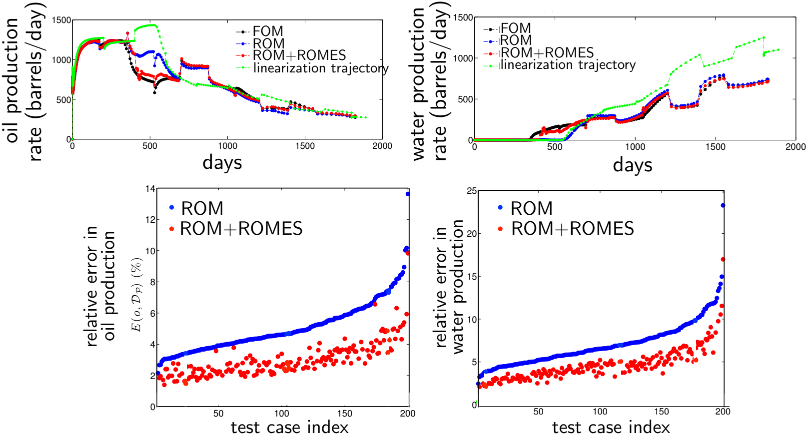


Permeability field with injection I_j and production P_j wells

- Two-phase oil–water system in porous medium (Darcy's law)
- 5×10^3 degrees of freedom
- **Inputs**: time-varying bottom-hole pressure (BHP) at injector wells
- **QoI**: oil/water production rates
- **ROM**: POD–TPWL with $|\mathcal{D}_{\text{ROM}}| = 3$
- **ROMES**: $|\mathcal{D}_{\text{ROMES}}| = 200$

Error: phase flow rates at production well

Error indicators: 168 application-specific quantities



+ ROMES correction significantly improves ROM prediction

Summary: ROM and ROMES

Reduce the FOM dimensionality and quantify the introduced uncertainty

1. Reduced-order model (ROM)
   - **Goal**: low-dim dynamical system that accurately represents FOM
   - **Approach**: unsupervised machine learning and projection
   + physics-based approximation
   + can preserve special problem structure
   + high speedups possible
2. Reduced-order model error surrogate (ROMES)
   - **Goal**: unbiased, low-variance statistical model of the ROM error
   - **Approach**: supervised machine learning (regression)
   + more useful than error bounds (not sharp)
   + quantifies ROM-induced epistemic uncertainty
   + enables rigorous integration with UQ

Questions?

ROM references:

- C, Barone, and Antil. Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction. Journal of Computational Physics, 330:693–734, 2017.
- C, Farhat, Cortial, and Amsallem. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. Journal of Computational Physics, 242:623–647, 2013.
- C, Farhat, and Bou-Mosleh. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. International Journal for Numerical Methods in Engineering, 86(2):155–181, April 2011.

ROMES references:

- Drohmann and C. The ROMES method for statistical modeling of reduced-order-model error. SIAM/ASA Journal on Uncertainty Quantification, 3(1):116–145, 2015.
- Trehan, C, and Durlofsky. Error estimation for surrogate models of dynamical systems using machine learning. Submitted to the International Journal for Numerical Methods in Engineering, 2017.
- Freno and C, Applying machine learning to statistically model the error in approximate solutions to parameterized nonlinear algebraic equations. In preparation, 2017.

Acknowledgments

- This research was supported in part by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000.

C., K. (2015).
Adaptive *h*-refinement for reduced-order models.
International Journal for Numerical Methods in Engineering, 102(5):1192–1210.

C., K., Barone, M., and Antil, H. (2017).
Galerkin v. discrete-optimal projection in nonlinear model reduction.
Journal of Computational Physics, 330:693–734.

C., K., Farhat, C., Cortial, J., and Amsallem, D. (2013).
The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows.
Journal of Computational Physics, 242:623–647.

Drohmann, M. and C., K. (2015).
The romes method for reduced-order-model uncertainty quantification.

SIAM/ASA Journal on Uncertainty Quantification, 3(1):116–145.

Everson, R. and Sirovich, L. (1995).
Karhunen–Loève procedure for gappy data.
Journal of the Optical Society of America A, 12(8):1657–1664.

Farhat, C., Geuzaine, P., and Brown, G. (2003).
Application of a three-field nonlinear fluid-structure formulation to the prediction of the aeroelastic parameters of an F-16 fighter.
Computers & Fluids, 32(1):3–29.

Freno, B. and C, K. (2017).
Applying machine learning to statistically model the error in approximate solutions to parameterized nonlinear algebraic equations.
In preparation.

Patera, A. T. and Rozza, G. (2006).

Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations.
MIT.

Trehan, S., C, K., and Durlofsky, L. J. (2017).
Error estimation for surrogate models of dynamical systems using machine learning.
arXiv preprint arXiv:1701.03240.